



The Trilinos Export Makefile System for Portable and Scalable Dependency Tracking

Roger Pawlowski, Roscoe Bartlett, and Jim Willenbring

Sandia National Laboratories
Albuquerque, NM

Trilinos User Group Meeting
October 31st, 2005



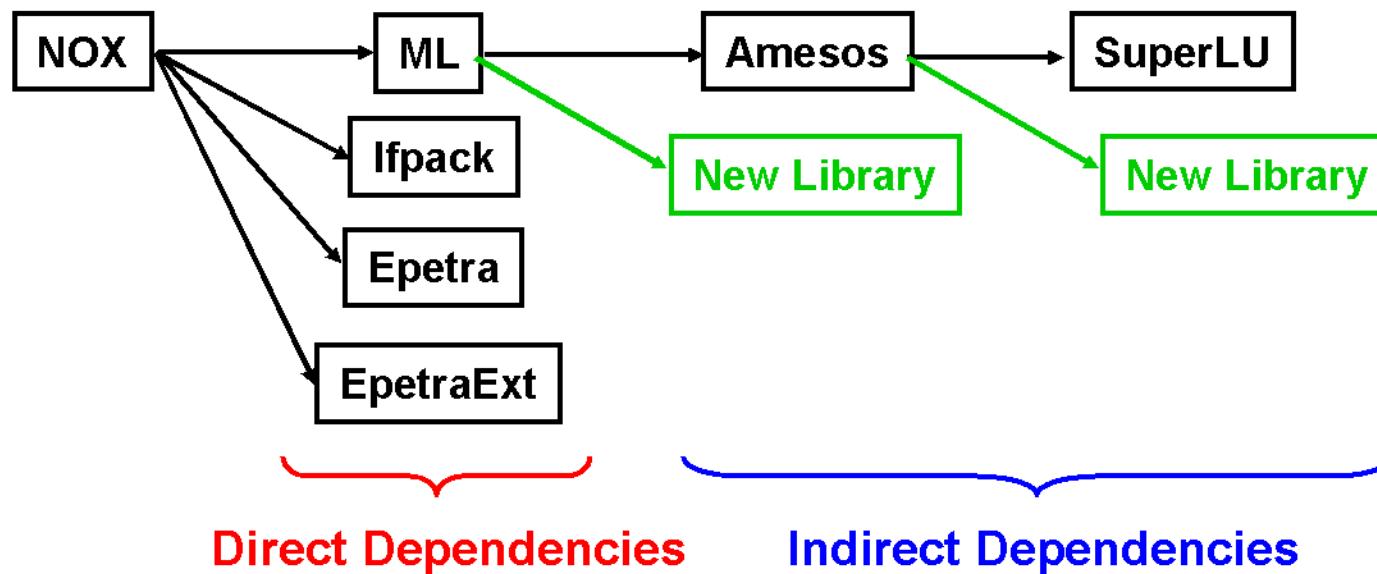
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.





Why Export Makefiles are Important

- The number of packages in Trilinos has exploded.
- As package dependencies (especially optional ones) are introduced, more maintenance is required by the top-level packages:



NOX either must:

- Account for the new libraries in its configure script (unscalable), or
- Depend on direct dependent packages to supply them through export makefiles.



Motivation continued

- Additionally, while Trilinos has had an internal export makefile system, nothing has existed to allow applications to access the export makefiles.
- This work:
 - Adapts the original Makefile.export files so that they can be installed in the prefix directory and used directly by applications!
 - Optionally allows users to clean up the build and link lines by removing redundant includes and libraries (gnumake dependent).

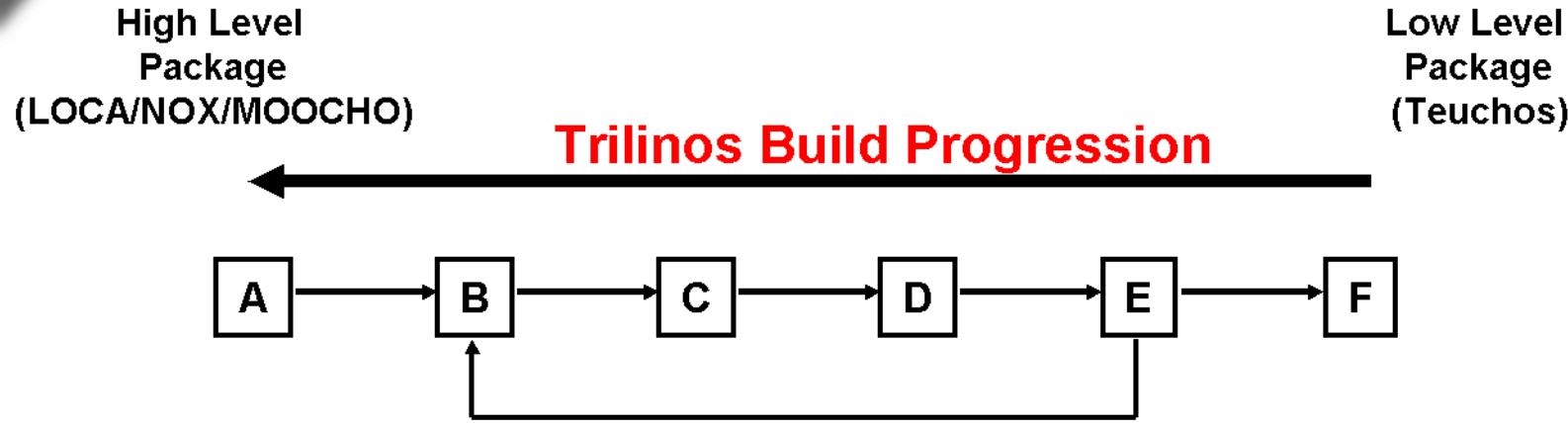


Requirements

- Portability – must run on all supported platforms with minimal additional requirements.
- Packages only **need account for direct dependencies**. They should know nothing about indirect dependencies.
- Circular dependencies must be supported.



Circular Dependencies

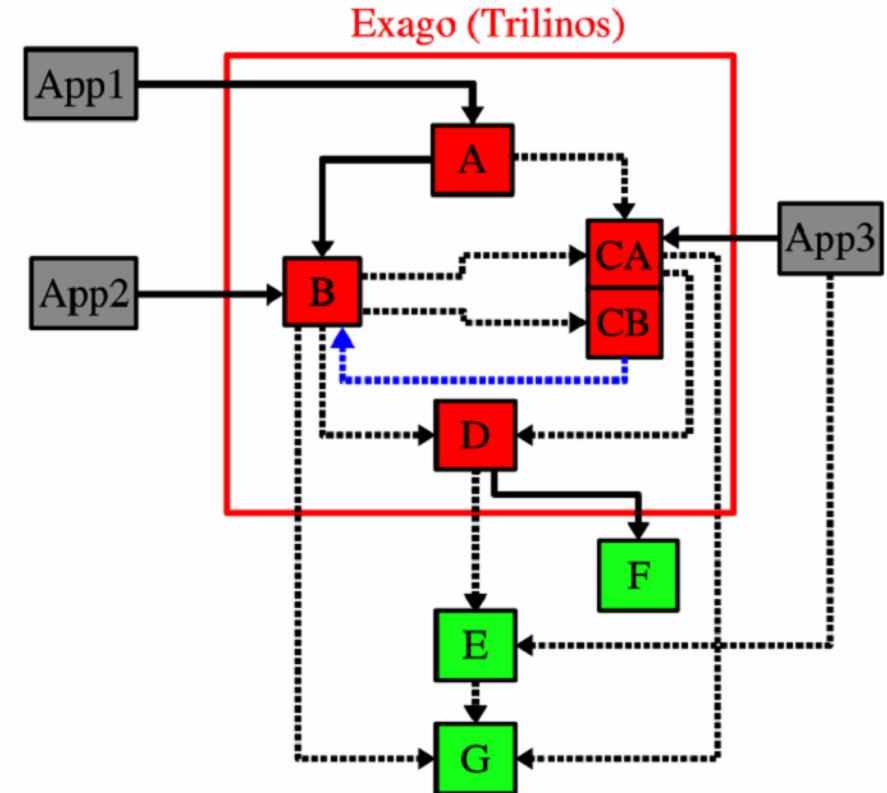


- Trilinos builds lower level libraries first.
- If there is a circular dependency → E depends on B, then the tests/examples in E that directly use code in B will fail to be built:
 - E requires libraries B, C, and D be built and linked in.
- The “all-libs” target allows a code to jump out of its build process and build ONLY the libraries of higher level packages.
- In the example above, after E’s library is built, and before it links its examples, B, C, and D’s libraries will be built.
- See [exago](#) or [aztecoo/thyra](#) for a working example of this process (We will implement library circular dependencies for NOX and ML soon).



Exago (“Export”)

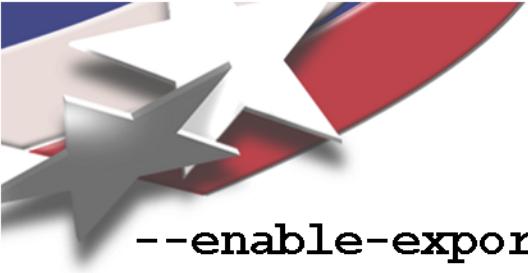
- A light-weight copy of the Trilinos build system.
- Provides a **fast-configuring/compiling** test-bed for the export makefile system.
- Demonstrated portability.
- Examples on how to link user applications to Trilinos using the Export makefiles.
- **Example using circular dependencies.**
- Contains a draft SAND report of how to implement the export makefile system (not finished yet).
- On software.sandia.gov:
 - setenv CVSROOT:ext:<username>@software.sandia.gov:/space/CVS-exago
 - cvs checkout exago





Overview

- Each package creates an export makefile named:
`Makefile.export.<package name>{.in} → Makefile.export.nox{.in}`
- This file provides the following variables to make:
`<PACKAGE NAME>_INCLUDES → NOX_INCLUDES`
`<PACKAGE NAME>_LIBS → NOX_LIBS`
- Each variable above should include **all package dependencies** it needs to link with an application.
- Any indirect dependencies should be brought in through the export files of other packages.
- Scripts create a file that contains compilers and build flags:
 - `Makefile.export.<package name>.macros`
- For circular dependencies, an extra make target must be defined called “all-libs”.



Flags to Support

--enable-export-makefiles

- Creates export makefiles in the install (prefix) directory.
- This option requires perl to be set in your path or defined with --with-perl=<perl executable>.
- Note that the export makefiles are always created and used in the build directory, but will not be installable without this option to change the paths.
- Found in the file "tac_arg_enable_export_makefiles.m4".

--with-gnumake

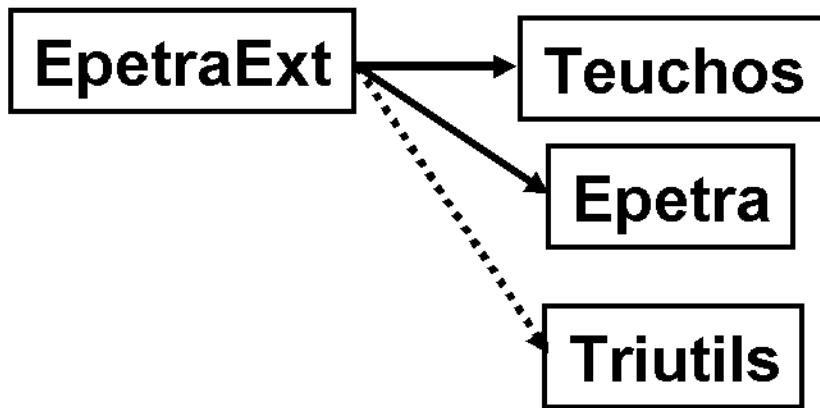
- Gnu's make has special functions we can use to eliminate redundant paths in the build and link lines.
- Enable this if you use gnu's version of make to build Trilinos.
- This requires that perl is in your path or that you have specified the perl executable with --with-perl=<perl executable>.
- Configure will check for the existence of the perl executable and quit with an error if it is not found.
- Most machines MUST build with gnumake enabled because the operating system can't support such long link lines!
- Found in the file "tac_arg_enable_export_makefiles.m4".

--with-perl

- Allows user to specify a perl executable. For example:
 --with-perl=/usr/bin/perl.
- Found in the file "tac_arg_with_perl.m4".



Example: EpetraExt



- Required dependency on Teuchos and Epetra
- Optional dependency on Triutils



Step 1: File Support

- Copy over the seven files into the “Trilinos/packages/<package>/config” directory.
 1. `string-replace.pl` – string replacement utility
 2. `strip_dup_inc_paths.pl` – removes duplicate include paths from makefile variable
 3. `strip_dup_libs.pl` – removes duplicate libraries from makefile variable
 4. `replace-install-prefix.pl` – Calls the above perl scripts on the eport makefile if installing into a directory
 5. `tac_arg_enable-export-makefiles.m4`
 6. `tac_arg_enable_can_use_package.m4`
 7. `tar_arg_with_perl.m4`



Step 2: Add calls to configure.ac file

```
# -----  
# Checks for Makefile.export related flags  
# -----  
TAC_ARG_ENABLE_EXPORT_MAKEFILES(yes)  
  
# -----  
# Checks for optional packages  
# -----  
TAC_ARG_ENABLE_CAN_USE_PACKAGE(epetraext,  
    triutils,  
    EPETRAEXT_TRIUTILS,  
    BUILD_TRIUTILS,  
    yes,  
    [Build triutils support in epetraext.],  
    [Build triutils support in epetraext. Can be  
overridden with --disable-epetraext-triutils.])  
  
AC_CONFIG_FILES([  
    Makefile  
    Makefile.export.epetraext  
    src/Makefile  
    test/Makefile
```

--enable-makfile-export
--enable-gnumake
--with-perl

--enable-epetra-triutils
Use implicit dependency: if "yes" then will look for:
--enable-triutils

still can be overridden with:
AC_DEFINE(HAVE_EPETRAEXT_TRIUTILS)
AM_CONDITIONAL(BUILD_TRIUTILS)

Help strings for the two flags:
--enable-epetraext-triutils
--enable-triutils

Step 3: Create Makefile.export.epetraext.in

```

## Required Dependency on teuchos
include @abs_top_builddir@/ /teuchos/Makefile.export.teuchos

## Required Dependency on Epetra
include @abs_top_builddir@/.. /epetra/Makefile.export.epetra

## Optional dependency on triutils
@BUILD_TRIUTILS_TRUE@ include @abs_top_builddir@/.. /triutils/Makefile.export.triutils

_EPETRAEXT_INCLUDES = -I@abs_top_builddir@/src -I@abs_top_srcdir@/src \
                     -I@abs_top_srcdir@/src/block \
                     -I@abs_top_srcdir@/src/btf \
                     -I@abs_top_srcdir@/src/coloring \
                     -I@abs_top_srcdir@/src/distdir \
                     -I@abs_top_srcdir@/src/inout \
                     -I@abs_top_srcdir@/src/matlab \
                     -I@abs_top_srcdir@/src/model_evaluator \
                     -I@abs_top_srcdir@/src/transform \
                     -I@abs_top_srcdir@/src/zoltan \
                     $(EPETRA_INCLUDES) \
                     $(TRIUTILS_INCLUDES) \
                     $(TEUCHOS_INCLUDES)

_EPETRAEXT_LIBS = \
@LDFLAGS@ -L@abs_top_builddir@/src -lepetraext \
$(EPETRA_LIBS) $(TRIUTILS_LIBS) $(TEUCHOS_LIBS) @LIBINTL@ @LAPACK_LIBS@ @BLAS_LIBS@ @FLIBS@

@USING_GNUMAKE_TRUE@EPETRAEXT_INCLUDES = $(shell @PERL_EXE@ \
    @abs_top_srcdir@/config/strip_dup_incl_paths.pl $_EPETRAEXT_INCLUDES)
@USING_GNUMAKE_TRUE@EPETRAEXT_LIBS = $(shell @PERL_EXE@ \
    @abs_top_srcdir@/config/strip_dup_libs.pl $_EPETRAEXT_LIBS) \
@USING_GNUMAKE_FALSE@EPETRAEXT_INCLUDES = $(EPETRAEXT_INCLUDES)
@USING_GNUMAKE_FALSE@EPETRAEXT_LIBS = $(EPETRAEXT_LIBS)

```

If using GNUMake then perl scripts remove duplicates

Configure creates the actual file!

Direct macro substitution only for autoconf variables!

Regular macros for xxx_INCLUDE and xxx_LIBS

Critical to remember flags that Trilinos sets!



Step 4: Edit Top-Level Makefile.am

Add the “all-libs” target for circular dependencies:

```
## #####  
## Target for circular dependencies  
## #####  
  
all-libs:  
    cd $(top_builddir)/src ; $(MAKE) all-libs
```

This target must also be added to any Makefile.am
in a directory that creates a library such as:

<package>/src



Step 4 continued: Edit Top-Level Makefile.am

Add the install-hook to call the perl scripts to change the directory paths of the installed Makefile.export files:

```
## #####  
## Export Makefile Installation  
## #####  
if USING_EXPORT_MAKEFILES  
  
install-exec-hook:  
    cp $(top_builddir)/Makefile.export.epetraext $(exec_prefix)/include/.  
    $(PERL_EXE) $(top_srcdir)/config/replace-install-prefix.pl \  
        --exec-prefix=$(exec_prefix) \  
        --my-export-makefile=Makefile.export.epetraext \  
        --my-abs-top-srcdir=@abs_top_srcdir@ \  
        --my-abs-incl-  
    dirs=@abs_top_builddir@/src:@abs_top_srcdir@/src:@abs_top_srcdir@/src/block:@abs_top_srcdir@/src/btf:@abs_top_srcdir@/src/coloring:@abs_top_srcdir@/src/distdir:@abs_top_srcdir@/src/inout:@abs_top_srcdir@/src/matlab:@abs_top_srcdir@/src/transform:@abs_top_srcdir@/src/zoltan \  
        --my-abs-lib-dirs=@abs_top_builddir@/src \  
        --dep-package-abs-  
builddirs=@abs_top_builddir@/../epetra:@abs_top_builddir@/../triutils  
    $(PERL_EXE) $(top_srcdir)/config/generate-makeoptions.pl $(top_builddir)/src/Makefile.epetraext >  
$(exec_prefix)/include/Makefile.export.epetraext.macros  
  
uninstall-hook:  
    rm -f $(exec_prefix)/include/Makefile.export.epetraext  
    rm -f $(exec_prefix)/include/Makefile.export.epetraext.macros  
  
else  
  
install-exec-hook:  
uninstall-hook:  
  
endif
```

Tells the perl scripts about include and lib paths

Generates the file Makefile.export.epetraext.macros



Step 5: Edit the EpetraExt/src/Makefile.am

Include the export makefile to access the EPETRAEXT_INCLUDES variable:

```
# -----
include $(top_builddir)/Makefile.export.epetraext

AM_CPPFLAGS = $(EPETRAEXT_INCLUDES)
```

Add the circular dependency target “all-libs”:

```
# -----
# Library Target for Circular Dependencies
#
if BUILD_TRIUTILS
BUILD_TRIUTILS_LIBS = cd $(top_builddir)/../triutils ; $(MAKE) all-libs
else
BUILD_TRIUTILS_LIBS =
endif

all-libs:
    $(MAKE) libepetraext.a
    $(BUILD_TRIUTILS_LIBS)
```



Step 6: Edit Test/Example Makefile.am to use the Export Makefiles

```
EXEEXT = .exe

noinst_PROGRAMS = MapColoring_test

MapColoring_test_SOURCES = \
$(srcdir)/cxx_main.cpp \
$(top_srcdir)/test/epetra_test_err.h

MapColoring_test_DEPENDENCIES = \
$(top_builddir)/src/libepetraext.a \
$(top_builddir)/../epetra/src/libepetra.a

include $(top_builddir)/Makefile.export.epetraext

if USING_GNUMAKE
EXPORT_LIBS = $(shell $(PERL_EXE) $(top_srcdir)/config/strip_dup_libs.pl
$EPETRAEXT_LIBS)
EXPORT_INC_PATH = $(shell $(PERL_EXE)
$(top_srcdir)/config/strip_dup_incl_paths.pl $EPETRAEXT_INCLUDES)
else
EXPORT_LIBS = $(EPETRAEXT_LIBS)
EXPORT_INC_PATH = $(EPETRAEXT_INCLUDES)
endif

AM_CPPFLAGS = $(EXPORT_INC_PATH)

MapColoring_test_LDADD = $(EXPORT_LIBS)
```

Include the export makefile

} **Strips Duplicates**

} **Allows Duplicates**



Application Usage (Trilinos Installed)

```
##  
## Application 1 is a non-autoconfed application  
  
TRILINOS_INSTALL_DIR = ../LIB  
  
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.epetraext.macros  
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.epetraext  
  
COMPILE_FLAGS = $(EPETRAEXT_CXXFLAGS) $(EPETRAEXT_DEFS) $(EPETRAEXT_CPPFLAGS) \  
    $(EPETRAEXT_INCLUDES)  
  
LINK_FLAGS = $(EPETRAEXT_LIBS)  
  
app1.exe: app1.o  
    $(EPETRAEXT_CXXLD) $(EPETRAEXT_CXXFLAGS) -o app1.exe app1.o $(LINK_FLAGS)  
  
app1.o:  
    $(EPETRAEXT_CXX) $(COMPILE_FLAGS) -c app1.cpp  
  
clean:  
    rm -f *.o app1.exe *
```

Include only the packages that you have DIRECT dependencies on



CXXLD, CXX, CXXFLAGS, DEFS, CPPFLAGS comes from macros file.
Guarantees Compatible Objects code when linking!



Application Usage (From Trilinos Build Dir)

```
##  
## Application 1 is a non-autoconfed application  
  
TRILINOS_HOME = /rabartl/PROJECTS/Trilinos.base/Trilinos  
TRILINOS_BUILD = /rabartl/PROJECTS/Trilinos.base/BUILDS/GCC-3.4.3/SERIAL_OPTIMIZED  
  
include Makefile.export.epetraext.macros  
include $(TRILINOS_BUILD)/packages/epetraext/Makefile.export.epetraext  
  
COMPILE_FLAGS = $(EPETRAEXT_CXXFLAGS) $(EPETRAEXT_DEFS) $(EPETRAEXT_CPPFLAGS) \  
$(EPETRAEXT_INCLUDES)  
  
LINK_FLAGS = $(EPETRAEXT_LIBS)  
  
app1.exe: app1.o  
    $(EPETRAEXT_CXXLD) $(EPETRAEXT_CXXFLAGS) -o app1.exe app1.o $(LINK_FLAGS)  
  
app1.o:  
    $(EPETRAEXT_CXX) $(COMPILE_FLAGS) -c app1.cpp  
  
clean:  
    rm -f *.o app1.exe *~
```

```
top_srcdir = $(TRILINOS_HOME)/packages/epetraext  
top_builddir = $(TRILINOS_BUILD)/packages/epetraext  
  
Makefile.export.epetraext.macros : $(top_builddir)/src/Makefile  
perl $(top_srcdir)/config/generate-makeoptions.pl \  
$(top_builddir)/src/Makefile.epetraext \  
> Makefile.export.epetraext.macros
```

**Build macros file
automatically!**



Current Status

- Trilinos Release 6.0 contains 13 packages that have been adapted to use the new export system:
Amesos, Anasazi, AztecOO, Belos, Epetra, Epetraext, Ifpack, ML, NOX, LOCA, Teuchos, Thyra, and Triutils.
- Ported to all required Trilinos platforms, but many have required the use of the –with-gnumake option.
- Contact Roger Pawlowski or Roscoe Bartlett for help in setting up your package.



Caveats

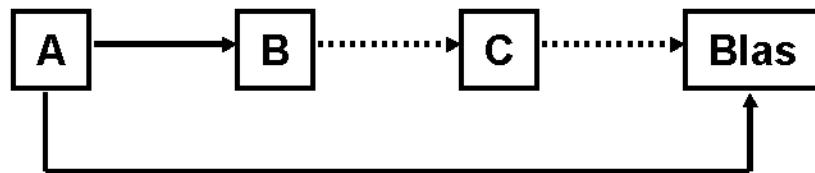
- Export makefiles are only for Trilinos libraries. Third party libraries still need to be added using –with-libs and –with-includes flags.



Extra Slides



Why we chose export makefiles and require gnumake.



A always depends on B and Blas.

B optionally depends on C.

C optionally depends on Blas.

This fact forces A to:

1. Know all about indirect dependencies
2. Put in configure options for Blas that depend on how indirect packages are configured.